Real-Time Rendering of Algebraic Surfaces

Csongor Csanád Karikó, Gábor Valasek
Eötvös Loránd University, Hungary

PROBLEM

We investigate the problem of robust and real-time rendering of algebraic surfaces. We show that expressing the intersection of the ray and the algebraic surface as a single univariate polynomial is not robust in practice, comparing results between monomial, Bernstein, Lagrange, and Chebyshev basis fits. We show that fitting multiple polynomials over subintervals, such as a unit length subdivision of the ray extent within the region of interest, improves robustness at a negligible performance cost.

The purpose of this work is to investigate the numeric stability of different polynomial bases when rendering algebraic surfaces in a finite subset of space. That is, we assume that $f(x, y, z)$ is a polynomial of known degree.

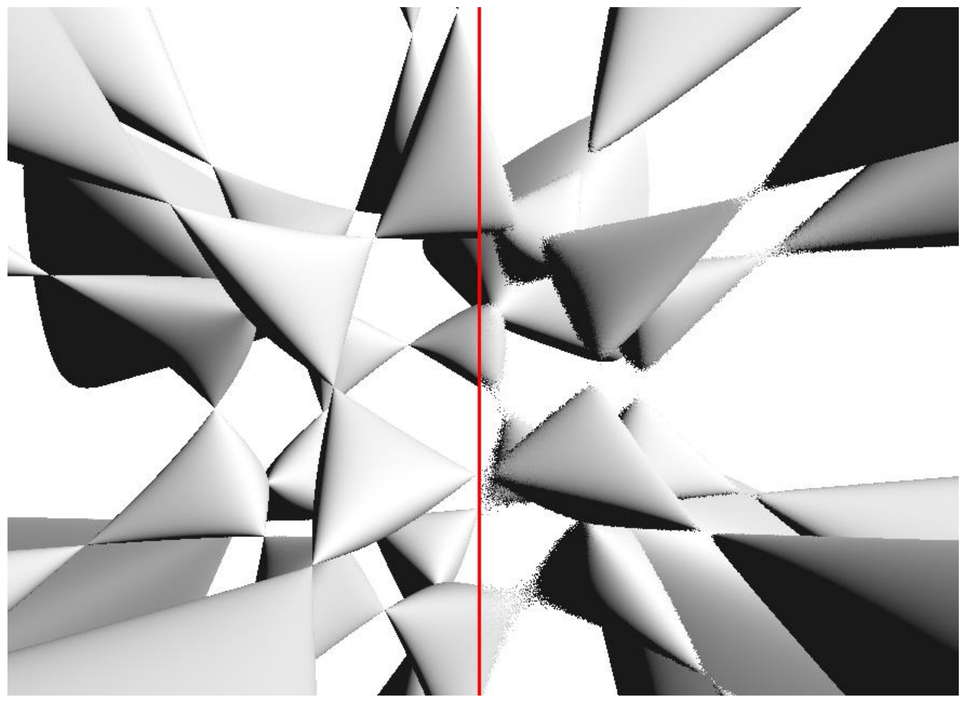
RELATED WORK

Rendering surfaces defined as isocontours of general f: $\mathbb{R}^3 \to \mathbb{R}$ mappings is a challenging task. Achieving high performance and robustness simultaneously is only viable for constrained subsets of such mappings, each requiring its own specialized algorithms. For example, if f is Lipschitz-continous with a known or easily computable Lipschitz-constant, it is possible to use sphere tracing [1,2] or one of its variants.

A family of methods involve fitting one or more polynomials to $f(t)$, reducing the problem to one dimensional root finding [3]. With this approach, there are multiple choices to be made surrounding the used polynomial basis, the fitting method, and the root finder. Rendering artifacts, may originate from inaccurate fitting, numerically unstable evaluation, and imprecise root finding.

OVERVIEW

In general, the rendering task is formulated as finding the smallest positive $t \in \mathbb{R}$ such that $f(\boldsymbol{o} + t \cdot \boldsymbol{d}) = 0$, where $\boldsymbol{o}$ is the origin of the ray and $\boldsymbol{d}$ is its direction. As evaluating f frequently can be performance heavy, we replace it with single variate polynomials fit over multiple segments of the ray function $r(t) = f(\boldsymbol{o} + t \cdot \boldsymbol{d})$. We use $N + 1$ samples for a surface of degree N, producing an exact fit. We used the Chebyshev-Lobatto grid on [0,1] for determining the sample positions. A polynomial is interpolated for the segment, then root finding is accomplished through simple ray marching. That is, we took equidistant steps along the ray until a sign change. If no root is found, we repeat the process for the next segment until we exit the bounding box or retrieve a root.



Direct ray marching (reference)

Single exact fit polynomial (Monomial basis)

We used the Barth sextic surface for illustration, defined as the zero contour of the function below:

$$f(x,y,z) = 4\left(\phi^{2x^2} - y^2\right)\left(\phi^{2y^2} - z^2\right)\left(\phi^{2z^2} - x^2\right) - (1 + 2\phi)(x^2 + y^2 + z^2 - 1)^2$$

Methodology

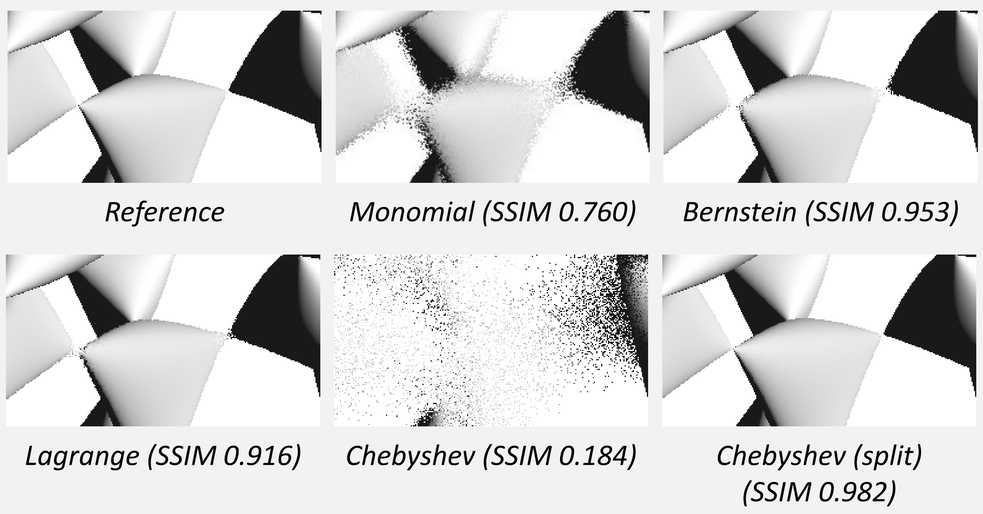
To assess the stability of the different polynomial bases, we rendered challenging algebraic surfaces by fitting a single polynomial per ray and then ray marching the fitted polynomial.

For performing the interpolation, we precomputed the QR decomposition of the interpolation matrix and solved the resulting system of linear equations in the shader with back substitution.

We used numerically stable methods for evaluating the polynomials. For monomial and Chebyshev basis polynomials, we used Horner's and Clenshaw's algorithms, respectively. Lagrange polynomials were evaluated with the barycentric evaluation formula while Bernstein polynomials with the de Casteljau algorithm.

We implemented these techniques in C++ using the Falcor system.

RESULTS



Reference | *Monomial (SSIM 0.760)* | *Bernstein (SSIM 0.953)*

Lagrange (SSIM 0.916) | *Chebyshev (SSIM 0.184)* | *Chebyshev (split) (SSIM 0.982)*

All bases exhibit rendering artifacts when rendered with only a single segment. Splitting the polynomial into multiple segments improves rendering accuracy significantly, at a cost of execution time.

Basis	Single segment	Multi segment
Monomial	*7.46 ms*	*9.62 ms*
Bernstein	*400.1 ms*	*401.96 ms*
Lagrange	*33.65 ms*	*33.2 ms*
Chebyshev	*10.33 ms*	*12.2 ms*
Direct ray march	*14.32 ms*	

Comparison of the render time of a single 1920×1080 *frame for the different bases. The step length for ray march is the same for all methods. Multi segment versions may split the ray into a maximum of 10 unit length segments The last row contains timings using ray marching on the original* $f(x, y, z)$ *function. Ray marching used a step length of 0.005 in both cases.*

The tests were carried out on a laptop with an AMD Ryzen 7 7840HS CPU and NVIDIA RTX 4060 laptop GPU. The reported performance figures are timing reports of the render dispatch taken in Nsight.

Converting algebraic surfaces to univariate polynomials along rays is a viable, albeit nontrivial way to render ray-surface intersections. In theory, sampling the surface function at $N + 1$ positions along the ray provides all the necessary points to interpolate $f(t)$ exactly. In practice, however, the fitting and evaluation of such polynomials is not accurate enough, regardless of the basis. For challenging surfaces, such as the Barth sextic, it is necessary to split the ray into multiple segments. The monomial and Chebyshev bases offer better performance than the reference ray march on the three-variable input. The quadratic complexity of the de Casteljau algorithm makes it not viable for real-time rendering.

AFFILIATIONS

ELTE EÖTVÖS LORÁND UNIVERSITY

EKÖP Egyetemi Kutatói Ösztöndíj Program

REFERENCES

[1] HART J. C.: Sphere Tracing: A Geometric Method for the An-tialiased Ray Tracing of Implicit Surfaces. The Visual Computer 12, 10(Dec. 1996), 527–545

[2] KALRA D., BARR A. H.: Guaranteed Ray Intersections with Implicit Surfaces. In Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques (July 1989), SIGGRAPH ’89,ACM, pp. 297–306.

[3] WINCHENBACH R., MÖLLER M., KOLB A.: Lipschitz-agnostic, efficient and accurate rendering of implicit surfaces. The Visual Computer (01 2024), 1–20.

Supported by the EKÖP-24 University Excellence scholarship program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation fund. Supported by ELTE Eötvös Loránd University, Budapest, Hungary.